Zooming-out on Higraph-based diagrams: Syntactic and Semantic Issues

Stuart Anderson [3] John Power [1,4] and Konstantinos Tourlas [2,5]

Division of Informatics, The University of Edinburgh, Edinburgh EH9 3JZ, United Kingdom

Abstract

Computing system representations based on Harel's notion of hierarchical graph, or higraph, have become popular since the invention of Statecharts. Such hierarchical representations support a useful filtering operation, called "zooming-out", which is used to manage the level of detail presented to the user designing or reasoning about a large and complex system. In the framework of (lightweight) category theory, we develop the mathematics of zooming out for higraphs with loose edges, formalise the transition semantics of such higraphs and conduct an analysis of the effect the operation of zooming out has on the semantic interpretations, as required for the soundness of reasoning arguments depending on zoom-out steps.

1 Introduction

Recent years have witnessed a rapid, ongoing popularisation of diagrammatic notations in the specification, modelling and programming of computing systems. Most notable among them are Statecharts [3], a notation for modelling reactive systems, and the Unified Modelling Language (UML) [9,10], a family of diagrammatic notations for object-based modelling. As the popularity of diagrammatic languages in computing and software engineering increases, so does the need of supporting best practice in terms of a sound theory accounting for the multitude of syntactic, semantic and pragmatic issues involved.

Key to the effectiveness of diagrammatic notations for design, modelling and reasoning is the wide range of manipulations they often support. In partic-

[1] This work has been done with the support of EPSRC grant GR/M56333 and a British Council grant, and the COE budget of STA Japan.
[2] Support EPSRC grant GR/N12480 and of the COE budget of STA Japan is gratefully acknowledged.
[3] Email: soa@dcs.ed.ac.uk
[4] Email: ajp@dcs.ed.ac.uk
[5] Email: kxt@dcs.ed.ac.uk

ular, the usability of diagrams representing realistic, complex systems crucially depends on one's ability to perform operations for re-organising, abstracting and filtering the information present in diagrams [8].

Both in system design and analysis it is very common that we want to control the level of detail. Often this is done by "chunking" some subsystem and representing it as a single object. For example, Figure 1 shows successive "chunkings" of a graph by hiding detail. Other such "complexity management" operations are more akin to "filtering" by eliminating detail uniformly throughout part of a diagram by applying some elimination criterion.

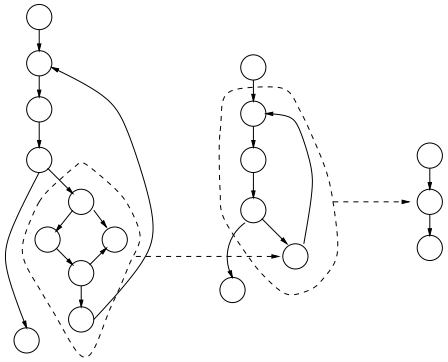


Fig. 1. Incremental "chunking" on a graph.

A central issue concerns the *semantic import* of such complexity management manipulations on diagrams. This is because one's purpose in performing such manipulations is to eliminate from a system's representation some detail which is deemed irrelevant to the task at hand (which may be a design or reasoning task), thereby simplifying one's task. It is therefore extremely important to precisely relate the semantics of the simplified diagram to that of the original, if completion of the task on the simple representation is to mean anything about the system represented by the original diagram.

The present paper develops a framework for the syntax and semantics of such a filtering operation on *higraphs* [4,5] extended with *loosely attached edges*. Higraphs (short for "hierarchical graphs") are themselves a simple extension of graphs allowing the containment of nodes inside other nodes, resulting in a hierarchy of "depth" among nodes and edges. Higraphs underlie the popular notation of Statecharts and the state diagrams of UML, and are typically interpreted as compact and economical representations of complex transition systems. The filtering operation we study, introduced briefly and motivated by Harel in [4] under the name of *zooming out*, exploits depth by eliminating detail below a certain level in the hierarchy.

Section 2 introduces the structure ("statics") of higraphs and their most common computational interpretation ("dynamics") as state-transition systems. In Section 3 we review the simplest form of zooming out on higraphs, studied extensively in [12], and summarise the detailed analysis in [1] pointing out its inadequacies wrt. the dynamics and the need for loosely attached edges.

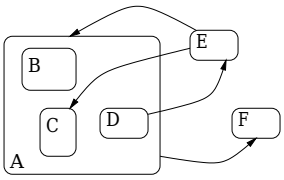


Fig. 2. A simple higraph.

The latter are formally introduced in Section 4, and in Section 5 we define a more refined notion of zooming out in the presence of such edges. Our main result, extending the work in [12,1], captures zooming out as a rewriting operation described by a pair of adjoint functors. The dynamics of higraphs with loosely attached edges is the subject of Section 6, where the notion of *run* is extended from ordinary (or "flat") transition systems to our higraphs. Finally, we provide an analysis of the issues arising in establishing a relation between the dynamics of a higraph-based system to the dynamics of its zoomed-out representation.

2 Higraphs

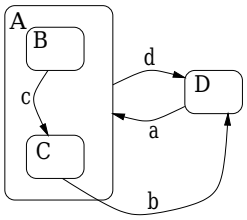
Higraphs, originally developed by Harel [4,5] as a foundation for Statecharts [3], are diagrammatic ("visual") objects which extend graphs by permitting spatial containment among the nodes. Figure 2 illustrates the pictorial representation of a simple higraph consisting of six nodes and four edges, with the nodes labelled B, C and D being spatially contained within the node labelled A. It is therefore common, and we shall hereafter adhere to convention, to call the nodes of a higraph *blobs*, as an indication of their pictorial representation as non-empty regions of the plane. A blob is called atomic if no other blobs are contained in it. The feature of spatial containment is often referred to as *depth*, leading to an expression of the relationship of higraphs to graphs in terms of Harel's "equation":

$$\text{higraphs} = \text{graphs} + \text{depth}\,^{6} \quad .$$

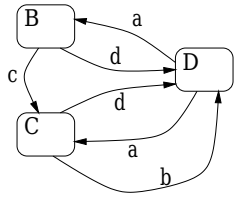
The main application of higraphs has been in the specification and visualisation of complex state-transition systems, manifested mainly in Statecharts and, more recently, in the state diagrams of UML [9]. In such applications, depth is used both as a conceptual device, in decomposing the overall system into meaningful subsystems, and as an economical and effective representation of *interrupts*. In terms of our example higraph in Figure 2, the edge emanat-

[6] Higraph is a term coined-up by Harel as short for *hierarchical graph*, but often used quite liberally to include several variants. The view taken here is that depth is the most distinguishing, definitive feature of higraphs. Harel's original definition includes an extra feature which he called *orthogonality* and which is not treated here. It is our conviction, supported by preliminary results outside the scope of the present paper, that orthogonality is, at least mathematically, best regarded as an extension to the basic, "depth-only" higraphs considered here.

ing from blob A may be regarded as a *higher-level* transition interrupting the operation of the subsystem comprising states (i.e. atomic blobs) B, C and D. When applied at multiple levels, depth therefore facilitates the concise representation of large systems by drastically reducing the number of edges required to specify the transition relation among states. Thus, for instance, the higraph



concisely represents the following transition system:



The reader should note that the transition interpretation of higraphs is more general than that of Statecharts. In Statecharts, a facility exists which, in terms of our example above, allows one to annotate either B or C as the *default* state within A. Such annotation forces the arrow from D to A to be a transition from D to the default state in A, thereby eliminating non-deterministic choice. Although such a device may readily be added to higraphs, we have chosen not to do so in the present paper for reasons of generality and simplicity of exposition.

Higraphs have also been used in extensions of other modelling notations, such as the Entity-Relationship (ER) diagrams popular in database analysis and design [4,5]. In that context, blobs denote sets (or "tables" of records of some particular type) whereas containment is directly interpreted as set inclusion.

2.1 Higraphs, formally

Our definition of higraph is based on Harel's [4] but uses posets (i.e. partially ordered sets) to capture the notion of depth and extends it also to the edges:

Definition 2.1 A *higraph* is a 5-tuple $(B, \leq_B, E, \leq_E, s, t)$, where B and E are respectively the sets of *blobs* and *edges*, $\leq_B$ is a partial order on B, $\leq_E$ is a partial order on E, and $s, t : E \to B$ are monotone functions giving, for each edge $e \in E$, its *source* blob $s(e)$ and target blob $t(e)$. □

Example 2.2 Figure 2 may be seen as the pictorial representation of a higraph with:

- blobs: $\{A, B, C, D, E, F\}$ where $B, C, D < A$

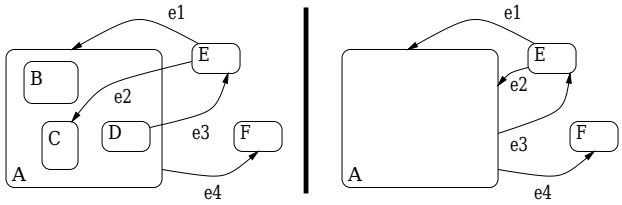


Fig. 3. Zooming out of a blob in a higraph

- edges: $\{e_1, e_2, e_3, e_4\}$ where $e_2 < e_1$
- $s(e_1) = E$, $t(e_1) = A$, $s(e_2) = E$, $t(e_2) = C$, and so on. □

Essentially, each higraph χ is a pair of "parallel" arrows in the category **Poset** of partially-ordered sets (posets) and monotone functions, which is to say, a graph in **Poset** [11]. Hereafter we shall denote higraphs with χ, χ', $\chi''\ldots$, and implicitly decompose them as $\chi = (s, t : E \to B)$, $\chi' = (s', t' : E' \to B')$ and so on, unless specifically indicated otherwise.

Other notions of "hierarchical graph" in the literature, such as clustered graphs [2], may be seen as instances of our definition of higraph. A (directed) clustered graph, for instance, is a higraph in which the poset B of blobs is a rooted tree, the poset E of edges is discrete, and edges may join only leaves of the tree. Thus, in general, higraphs not only account for clustering, but also allow one to represent relationships or transitions between clusters by means of edges.

3 Naive zooming out

We base our analysis on the simplest, and most frequently occurring in practice, instance of a zooming-out operation on higraphs: the selection of a single blob and the subsequent removal from view of all structure (blobs and edges) contained in it. An example, in which edges are conveniently shown labelled for ease of reference, is illustrated in the transition from the left to the right half of Figure 3. Notice, in particular, how the edges attached to the blobs contained in A are subsequently fixed to A.

This filtering operation on higraphs is introduced, albeit briefly and informally, and justified in terms of its practical significance, in [3,4]. In [12] this operation on higraphs is formalised in a category-theoretic framework, and subsequently generalised in [11].

Under the usual transition system interpretation of higraphs, the above simple, almost naive, notion of zooming out can introduce profound inconsistencies between the two views of the represented system. This issue, which is discussed at length in [1], arises because the specified dynamic behaviour is inferred from the representing higraph by considering paths:

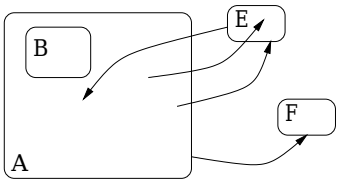
Definition 3.1 A *path* of length n in a higraph $(s, t : E \to B)$ is a sequence $\langle e_0, \ldots, e_{n-1} \rangle$ of edges in E such that $t(e_i) \leq_B s(e_{i+1})$ or $s(e_{i+1}) \leq_B t(e_i)$. A

path is said to include a blob b if b occurs as either the source or target of at least one edge in the path. □

Consider for instance the path consisting of edges e_2 and e_3 in the right-hand side of Figure 3. This sequence of transitions starting from E is, however, impossible in the original higraph (left-hand side). We describe this situation by saying that not all paths in the zoomed-out version of a higraph are necessarily *reflected* by paths in the original higraph. The unfortunate consequence is that reasoning about the represented system based on its higher-level, zoomed-out representation can be greatly complicated and, potentially, misguided. What one requires as a minimum is the ability to distinguish visually exactly those paths in a zoom-out of χ which are certain, i.e. guaranteed to be reflected by paths in the original higraph χ, from those which are not.

4 Higraphs with loosely attached edges

The analysis in [1] supports a solution to this problem which was proposed, albeit briefly and informally, by Harel in [4]. The solution requires a mild extension to higraphs which permits edges to be "loosely" attached to nodes, the four possibilities being illustrated in



An edge such as the one attached to the contours of A and F is called *firm*. The remaining three are *non-firm*.

In Figure 3 we observed that the sequence $\langle e_2, e_3 \rangle$ in the right-hand side is *not reflected* by a path in the original higraph (left-hand side). In the context of loose higraphs this deficiency is rectified. In Figure 4 we have the original higraph and its zoomed version as a loose higraph. The intuition is that we are no longer certain that the sequence $\langle e_2, e_3 \rangle$ is a connected path in the transition semantics for the original higraph. By contrast, the sequence $\langle e_1, e_3 \rangle$ is reflected by a path in the original, as e_1 implies edges to all the sub-blobs of blob A. Similarly, $\langle e_2, e_4 \rangle$ is also a certain path (i.e. reflected in the original).

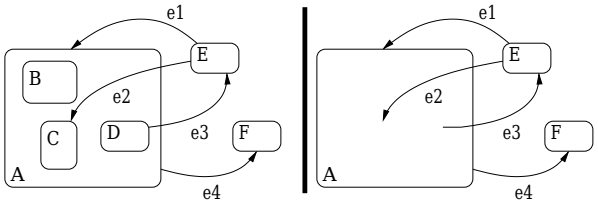


Fig. 4. Zooming out with loose edges

4.1 Formalisation of loosely attached edges

Formally, we cast such an extended higraph with blobs B as an ordinary one having the same edges but containing two distinct copies $\langle 0, b\rangle$ and $\langle 1, b\rangle$ of each $b \in B$, tagged with 0's and 1's. In the pictorial representation of such extended higraphs the convention is that blobs tagged with 0 are not shown at all and that, for instance, an edge with target of the form $\langle 0, b\rangle$ has its endpoint lying *inside* the contour picturing b.

In this setting, one stipulates that $\langle 0, b\rangle \leq \langle 1, b\rangle$ for all b, and moreover that $\langle 1, b\rangle \leq \langle 0, b'\rangle$ whenever $b < b'$, to capture the intuition underlying the pictorial representation of higraphs with loose edges [7] .

Definition 4.1 Given a poset $(A, \leq_A)$ define poset $A^\dagger$ to have underlying set $\{0, 1\} \times A$, ordered by:

- $\langle 0, a\rangle \leq \langle 1, a\rangle$ for all $a \in A$
- $\langle 1, a\rangle < \langle 0, a'\rangle$ whenever $a <_A a'$. □

Using this auxiliary poset structure we can now make precise the definition of a higraph with loosely attached edges:

Definition 4.2 A *higraph with loosely attached edges* is a higraph ϕ of the form $s, t : E \to B^\dagger$. The poset B will be referred to as the *underlying poset of blobs* of ϕ.

For brevity we shall hereafter abuse terminology and refer to higraphs with loosely attached edges as *loose higraphs*.

5 Zooming-out on loose higraphs

To capture the notion of selecting a blob in a loose higraph we introduce the following:

Definition 5.1 A *pointed loose higraph* $\phi_\star$ consists of a loose higraph ϕ, say with underlying poset of blobs B, together with a distinguished element $\langle 0, p\rangle \in B^\dagger$, called the *point* of $\phi_\star$. (This also selects an element p of B which may also be called the point, when no confusion arises.) □

Definition 5.2 A morphism from a pointed higraph $s, t : E \to B^\dagger$ with point $\langle 0, p\rangle$, to a pointed higraph $s', t' : E' \to B'^\dagger$ with point $\langle 0, p'\rangle$ consists of three monotone functions $m_E : E \to E'$, $m_B : B \to B'$ and $m^\dagger : B^\dagger \to B'^\dagger$ such that

- $m^\dagger \circ s = s' \circ m_E$ and $m^\dagger \circ t = t' \circ m_E$ (i.e. the pair $\langle m_E, m^\dagger\rangle$ is a morphism of ordinary higraphs [12]);

[7] A similar definition in [1] uses a simpler partial order on the tagged blobs. The more sophisticated order used here seems to better capture subtle aspects of the pictorial intuition which are salient to zooming.

- $m_B \circ \pi_1 = \pi'_1 \circ m^\dagger$, where $\pi_1 : B^\dagger \to B$ is the monotone function mapping each $\langle i, b\rangle \in B^\dagger$ to $b \in B$, and similarly for π'_1; and
- $m^\dagger(\langle 0, p\rangle) = \langle 0, p'\rangle$, i.e. points are preserved. □

Morphisms of pointed, loose higraphs compose component-wise and have obvious identities. Thus one has a category $\mathcal{LH}_\star$ of pointed loose higraphs.

With $\mathcal{LH}_{\star,min}$ we shall denote the (full) subcategory of $\mathcal{LH}_\star$ consisting of all pointed loose higraphs in which the point is minimal wrt. the partial order on $B^\dagger$. Now, the operation of zooming out may be viewed as a function Z from the objects $\mathcal{LH}_\star$ to those of $\mathcal{LH}_{\star,min}$ since, in essence, it reduces the point (selected blob) of $\phi_\star$ to a minimal point in $Z(\phi_\star)$:

Definition 5.3 Let $\phi_\star$ be a pointed loose higraph with $\phi = s, t : E \to B^\dagger$ and point $\langle 0, p\rangle \in B^\dagger$. Formally, $Z(\phi_\star)$ is determined by the following data:

- underlying poset of blobs: $B' \stackrel{\text{def}}{=} B \setminus \{b \mid b < p\}$, partially ordered according to the following rules:
  - $b'_1 \leq b'_2$ whenever $b'_1 \leq_B b'_2$
  - $p \leq b'$ whenever there is $b_0 \in B$ such that $b_0 \leq_B p$ and $b_0 \leq_B b'$.
- edges: E, with the source and target functions being $q \circ s$ and $q \circ t$ respectively, where $q : B^\dagger \to B'^\dagger$ is the monotone function mapping each $\langle i, b\rangle \not< \langle 1, p\rangle$ to $\langle i, b\rangle$ and each $\langle i, b\rangle < \langle 1, p\rangle$ to $\langle 0, p\rangle$;
- point: $\langle 0, p\rangle \in B'^\dagger$. □

Our main result on zooming-out in the presence of loosely attached edges parallels the result in [12] for ordinary higraphs. It is formulated in terms of an adjunction [7], the universal property of which means intuitively that the essence of Z is to turn the point of $\phi_\star$ to a minimal point, and it does so "least descriptively" wrt. the structure of $\phi_\star$.

Theorem 5.4 *The function Z extends to a functor from $\mathcal{LH}_\star$ to $\mathcal{LH}_{\star,min}$ which is left adjoint to the inclusion functor $I : \mathcal{LH}_{\star,min} \to \mathcal{LH}_\star$.*

Proof. *(Sketch) Consider $\phi_\star$ where $\phi = s, t : E \to B^\dagger$ and has point $\langle 0, p\rangle$. The unit $\eta_{\phi_\star} : \phi_\star \to I(Z(\phi_\star))$ of the adjunction has the following components:*

- *on edges, the identity* $\mathrm{id}_E : E \to E$
- *on blobs, the (monotone) function mapping each $b \not< p$ to b and each $b < p$ to p; and*
- *$q : B^\dagger \to B'^\dagger$, where q is exactly as in Definition 5.3.*

It is not hard to show that each $\eta_{\phi_\star}$ is indeed a morphism in $\mathcal{LH}_\star$. Given any other morphism $m : \phi_\star \to I(\phi''_\star)$ in the same category, consider the morphism $\hat{m} : Z(\phi_\star) \to \phi''_\star$ with components

- $\hat{m}_E = m_E$
- *$\hat{m}_B$ mapping each $b \in B'$ to $m_B(b)$; and*
- *$\hat{m}^\dagger$ given by $\langle i, b\rangle \mapsto m^\dagger(\langle i, b\rangle)$*

Calculation now reveals that $\hat{m}$ is the unique one such that $I(\hat{m}) \circ \eta_{\phi_\star} = m$.$\Box$

6 Dynamics of loose higraphs

In this section we make precise the "dynamics", i.e. transition semantics, of loose higraphs by introducing a notion of *run* akin to similar notions in use with ordinary ("flat") transition systems. A run is essentially a sequence of transitions together with the states through which the system is taken by performing the transitions. In ordinary transition systems, the sequence of states traversed is implicit in the notion of path (i.e. connected sequence of transitions). In higraphs, where higher-level edges are taken to imply lower-level ones, the notion of path no longer provides adequate state information, hence the need for a notion of "run".

At a glance, a run through a loose higraph with blobs B and edges E is a sequence of the form

$$\langle i_0, b_0 \rangle \xrightarrow{e_1} \langle i_1, b_1 \rangle \xrightarrow{e_2} \ldots \xrightarrow{e_n} \langle i_n, b_n \rangle$$

subject to conditions. Here, the b_j's are blobs (i.e. states of the system), the e_k's are edges and each i_j is either 0 or 1. Each pair $\langle i_j, b_j \rangle$ may be called a *configuration* of the system [8], whose intuitive meaning is to specify the current state of the system at two different levels of precision:

- a configuration $\langle 1, b \rangle$ means that all substates of the state b (including b itself) are candidates for being the current state of the system, but we lack precise information as to which substate is actually current.
- a configuration $\langle 0, b \rangle$ means that only *one* of the *proper* substates of b (i.e. excluding b itself) is the current state, and that we lack any precise information as to exactly which that substate is.

This interpretation is captured in the following partial order on configurations:

- $\langle 0, b \rangle \sqsubseteq \langle 1, b \rangle$ for all b; and
- $\langle 1, b \rangle \sqsubseteq \langle 1, b' \rangle$ whenever $b \leq b'$ in B.

Notice that this "information ordering" on configurations is different from the order on $B^\dagger$ above, reflecting intuition regarding the dynamics rather than static spatial containment in a picture.

We are now in position to present the full definition of a run:

Definition 6.1 A *run* through a pointed, loose higraph $\phi_\star$, where $\phi = s, t : E \to B^\dagger$, is a sequence of the form

$$\langle i_0, b_0 \rangle \xrightarrow{e_1} \langle i_1, b_1 \rangle \xrightarrow{e_2} \ldots \xrightarrow{e_n} \langle i_n, b_n \rangle$$

[8] Readers should not confuse our use of the term "configuration" with its use in the literature on Statecharts, as e.g. in [6].

subject to the following conditions:

(i) for all $1 \leq j \leq n$, $\langle i_j, b_j \rangle \sqsubseteq t(e_j)$ and $\langle i_{j-1}, b_{j-1} \rangle \sqsubseteq s(e_j)$; and

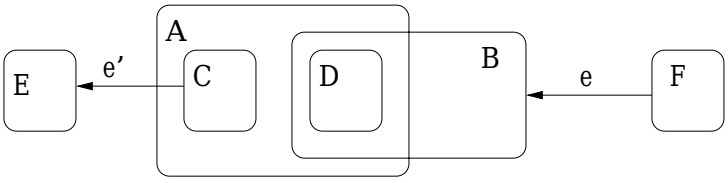
(ii) $\pi_0(t(e_j)) = 1$ or $\pi_0(s(e_{j+1})) = 1$ for all $1 \leq j < n$, where π_0 is the projection mapping each $\langle i, b \rangle \in B^\dagger$ to i. □

As an example, consider the right-hand side of Figure 4. While $\langle 1, E \rangle \xrightarrow{e_1} \langle 1, A \rangle \xrightarrow{e_4} \langle 1, F \rangle$ is a run, the sequence $\langle 1, E \rangle \xrightarrow{e_1} \langle 1, A \rangle \xrightarrow{e_3} \langle 1, E \rangle$ isn't because it violates the first condition. However $\langle 1, E \rangle \xrightarrow{e_1} \langle 0, A \rangle \xrightarrow{e_3} \langle 1, E \rangle$ is a run, as is $\langle 1, E \rangle \xrightarrow{e_1} \langle 0, A \rangle \xrightarrow{e_4} \langle 1, F \rangle$. But $\langle 1, E \rangle \xrightarrow{e_2} \langle 0, A \rangle \xrightarrow{e_3} \langle 1, E \rangle$ is not, as it violates the second condition.

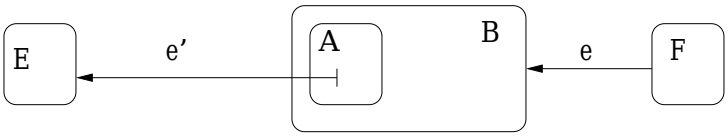
6.1 On relating the runs of $\phi_\star$ and $Z(\phi_\star)$

In order to support reasoning about higraph-based systems in the presence of zooming, one needs to relate the runs of a pointed loose higraph $\phi_\star$ to those of its zoom-out $Z(\phi_\star)$. In particular, one is interested in knowing which of the runs through the latter may be associated with runs through the former.

To see that there may be runs in $Z(\phi_\star)$ which are not reflected by runs in $\phi_\star$, consider for instance $\phi_\star$ to be the following loose higraph



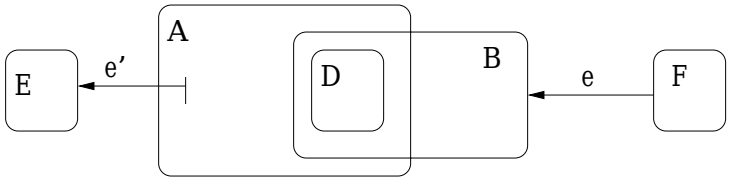
in which the point is A. Its zoom-out, according to Definition 5.3, is the loose higraph



Now one observes that while $\langle 1, F \rangle \xrightarrow{e} \langle 0, A \rangle \xrightarrow{e'} \langle 1, E \rangle$ is a run through the latter, it is not a run through the original higraph.

Thus one may

- seek constraints on the form of $\phi_\star$ which guarantee that every run through its zoom-out maps to a run in $\phi_\star$. For instance, the undesirable situation in the preceding example arises because the point, blob A, has a non trivial intersection (blob C) with another blob (B). The class of higraphs which do not feature such non-trivial intersections is an important one, as they occur commonly in practice, particularly in Statechart applications.
- investigate alternative definitions of zooming which agree with the one presented here when $\phi_\star$ does not feature non-trivial intersections. For instance, one may consider



to be an alternative zoom-out of our example above.

We are currently investigating both directions of research, together with a refinement of our notion of run into "certain" or "must be reflected" runs and "may be reflected" runs.

References

[1] Stuart Anderson, John Power, and Konstantinos Tourlas. Reasoning in higraphs with loose edges. In *Proceedings of the 2001 IEEE Symposia on Human-Centric Computing Languages and Environments*, pages 23–29. IEEE Computer Society Press, September 2001.

[2] P. Eades, Q. Feng, and H. Nagamochi. Drawing clustered graphs on an orthogonal grid. *J. Graph Algorithms Appl.*, 3(4):3–29, 1999.

[3] David Harel. Statecharts: A visual approach to complex systems. *Science of Computer Programming*, 8(3):231–275, 1987.

[4] David Harel. On visual formalisms. *Communications of the ACM*, 31(5):514–530, 1988.

[5] David Harel. On visual formalisms. In J. Glasgow, N.H. Narayanan, and B. Chandrasekaran, editors, *Diagrammatic Reasoning: Cognitive and Computational Perspectives*, pages 235–272. AAAI Press/The MIT Press, 1995.

[6] David Harel and Amnon Naamad. The STATEMATE semantics of Statecharts. *ACM Transactions on Software Engineering Methodology*, 5(4), October 1996.

[7] Saunders MacLane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer-Verlag, 1971.

[8] Bonnie M. Nardi. *A Small Matter of Programming: Perspectives on End-User Computing*. MIT Press, 1993.

[9] OMG ad/99-06-08 (Part 3). *UML Notation Guide version 1.3*, 1999.

[10] Rob Pooley and Perdita Stevens. *Using UML*. Addison Wesley, 1999.

[11] John Power and Konstantinos Tourlas. An algebraic foundation for graph-based diagrams in computing. In *Proceedings of the 17th Conference on the Mathematical Foundations of Programming Semantics (MFPS)*, 2001.

[12] John Power and Konstantinos Tourlas. An algebraic foundation for higraphs. In L. Fribourg, editor, *Proceedings of the 15th Annual Conference of the European Association for Computer Science Logic (CSL)*, volume 2142 of *Lecture Notes in Computer Science*, pages 145–159. Springer-Verlag, September 2001.